

# An ADMM Algorithm for Clustering Partially Observed Networks \*

N. S. Aybat<sup>†</sup>

S. Zarmehri<sup>‡</sup>

S. Kumara<sup>§</sup>

## Abstract

Community detection has attracted increasing attention during the past decade, and many algorithms have been proposed to find the underlying community structure in a given network. Many of these algorithms are based on modularity maximization, and these methods suffer from the resolution limit. In order to detect the underlying cluster structure, we propose a new convex formulation to decompose a partially observed adjacency matrix of a network into low-rank and sparse components. In such decomposition, the low-rank component encodes the cluster structure under certain assumptions. We also devise an alternating direction method of multipliers with increasing penalty sequence to solve this problem; and compare it with **Louvain** method, which maximizes the modularity, on some synthetic randomly generated networks. Numerical results show that our method outperforms **Louvain** method on the randomly generated networks when variance among cluster sizes increases. Moreover, empirical results also demonstrate that our formulation is indeed tighter than the robust PCA formulation, and is able to find the true clustering when the robust PCA formulation fails.

## 1 Introduction

Community detection or clustering is one of the most important topics in network science [18]. A cluster is defined loosely as a group of nodes which are more densely connected with each other than with nodes in the other groups of the network. Many clustering algorithms have been proposed to identify the underlying community structure in a given network. The goodness of the identified communities can be evaluated by a quality function [9]. Modularity is the most popular quality function [9] which was introduced by Girvan and Newman [19]. It is assumed that a higher modularity value indicates a better community structure. Although this is not always true, it has formed the motivation for developing many algorithms based on modularity maximization [9]. In particular, given a partition of nodes, modularity is the sum of values, each corresponding to a group in the partition. Hence, in modularity maximiza-

tion, one searches for the best partition, which is equivalent to looking for the best trade-off between the number of groups in the partition and their corresponding values; therefore, modularity maximization is an NP-complete problem [6] and the algorithms are only able to find a good approximation to the global solution. Recently, it has been shown that modularity maximization has some problems for large networks [15]. One major problem is due to *resolution limit* [10]. Many modularity based algorithms tend to merge smaller clusters with bigger ones even when the small size cluster is a clique, and it is connected to a larger cluster by a single edge [10, 15]. This problem arises from the definition of modularity and particularly from the assumption of its null model that each node can interact with any other node in the network [9]. If there are two communities with sufficiently small sizes (and hence small degrees), the expected number of edges between them for the null model is small. In this case, even the existence of a single edge between the two communities can merge them together [9]. Moreover, as discussed in [10], the partition corresponding to the highest modularity may not be correlated with the underlying unknown community structure. Indeed, there are some instances of real networks [10] and benchmark graphs [15] such that the modularity maximization fails to properly identify the community structure. Recently, it has been shown that the modularity maximization problem can have different local maxima which are structurally different but have high modularity values [11]. These solutions may disagree on many community structure properties such as the distribution of cluster sizes. This kind of disagreement may have serious impact on real world networks such as metabolic networks [11].

The need to find an accurate community structure motivated us to develop a new model, which depends more on the network structure and less on the quality function, together with an algorithm to solve it for community detection. Our method is based on convex optimization, and is inspired by the work in [7]. Suppose we have a data matrix  $D \in \mathbb{R}^{m \times n}$  which is a summation of a *low rank* matrix  $\tilde{L}$  and a *sparse* matrix  $\tilde{S}$ , i.e.,  $D = \tilde{L} + \tilde{S}$ . Consider the following convex optimization problem:

$$(1.1) \quad (L_\rho^*, S_\rho^*) \in \underset{L, S \in \mathbb{R}^{m \times n}}{\operatorname{argmin}} \{ \|L\|_* + \rho \|S\|_1 : D = L + S \},$$

\*For all questions please contact the first author. Partially supported by NSF grant CMMI-1400217.

<sup>†</sup>IME Dept. Penn State University. Email:nsa10@psu.edu.

<sup>‡</sup>IME Dept. Penn State University. Email:sxz155@psu.edu.

<sup>§</sup>IME Dept. Penn State University. Email:skumara@psu.edu.

where  $\|Z\|_* := \sum_{i=1}^{\text{rank}(Z)} \sigma_i(Z)$  denotes the nuclear norm of  $Z \in \mathbb{R}^{m \times n}$ , i.e., sum of singular values of its argument, and  $\|Z\|_1 = \sum_{i=1}^m \sum_{j=1}^n |Z_{ij}|$ . It has been shown in [7] that under some technical conditions on  $\bar{L}$  and  $\bar{S}$ , problem in (1.1) has a unique solution  $(L_\rho^*, S_\rho^*)$  such that  $(L_\rho^*, S_\rho^*) = (\bar{L}, \bar{S})$  with very high probability for  $\rho = 1/\sqrt{\max\{m, n\}}$ .

Suppose we are given an undirected network  $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ , where  $\mathcal{N} = \{1, \dots, n\}$  and  $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$  denote the set of nodes and edges, respectively. Suppose there are  $r \ll n$  communities in  $\mathcal{G}$ , and  $\mathcal{N}_\ell \subset \mathcal{N}$  denotes the subset of nodes in community- $\ell$  for  $1 \leq \ell \leq r$ . We assume that every node belongs to *exactly one* community, i.e.,  $\bigcup_{\ell=1}^r \mathcal{N}_\ell = \mathcal{N}$  and  $\mathcal{N}_{\ell_1} \cap \mathcal{N}_{\ell_2} = \emptyset$  for all  $\ell_1 \neq \ell_2$ . Let  $D \in \mathbb{R}^{n \times n}$  denote the node-node incidence matrix of  $\mathcal{G}$  such that  $D_{ii} = 1$  for all  $i \in \mathcal{N}$ ,  $D_{ij} = 1$  if either  $(i, j) \in \mathcal{E}$  or  $(j, i) \in \mathcal{E}$ , and  $D_{ij} = 0$  otherwise. Our idea is to decompose  $D$  into a low rank matrix and a sparse matrix to recover the underlying community structure in  $\mathcal{G}$ . Here we discuss that such decomposition is feasible under the assumption that the number of node pairs *not connected* by an edge in each cluster and the number of edges connecting two different clusters are both *small* for the underlying community structure in  $\mathcal{G}$ . To motivate the upcoming discussion, first consider the scenario where the subgraph of  $\mathcal{G}$  restricted to  $\mathcal{N}_\ell$  is a clique for all  $\ell = 1, \dots, r$ , and there is no inter-community edge in  $\mathcal{E}$ , i.e.,  $\mathcal{N}_{\ell_1} \times \mathcal{N}_{\ell_2} \subset \mathcal{E}^c$  for all  $\ell_1 \neq \ell_2$  and  $\mathcal{E}^c$  denotes the complement of  $\mathcal{E}$  in  $\mathcal{N} \times \mathcal{N}$ . Clearly,  $D$  is a block diagonal matrix with each block on the diagonal consisting of all ones and the off-diagonal blocks consisting of all zeros – from now on we refer to such matrices as block diagonal matrix of ones (BDO). Note that  $D$  is a low-rank matrix such that  $\text{rank}(D) = r$  and  $\lambda_\ell(D) = |\mathcal{N}_\ell|$  for all  $1 \leq \ell \leq r$ , where  $\{\lambda_\ell(D)\}_{\ell=1}^r$  denotes the non-zero eigenvalues of  $D$ . Hence,  $D = \bar{L} + \bar{S}$  such that low-rank component  $\bar{L} = D$  and sparse component  $\bar{S} = \mathbf{0}_n$ , where  $\mathbf{0}_n \in \mathbb{R}^{n \times n}$  is the matrix of zeros.

Now consider a more realistic scenario where for any  $\ell \in \{1, \dots, r\}$ , a *small* number of node pairs from  $\mathcal{N}_\ell$  may not be connected by an edge, and for any  $\ell_1 \neq \ell_2$  there may be a *small* number of edges with one end in  $\mathcal{N}_{\ell_1}$  and the other in  $\mathcal{N}_{\ell_2}$ , i.e., the clusters may not be cliques, and there can be inter-cluster edges. Given  $\mathcal{G}$  with a non-overlapping community structure  $\{\mathcal{N}_\ell\}_{\ell=1}^r$ , we define  $\bar{L} = D - \bar{S}$  and  $\bar{S}$  such that

$$(1.2) \quad \bar{S}_{ij} = \begin{cases} -1, & \text{if } (i, j) \notin \mathcal{E}, (j, i) \notin \mathcal{E}, \text{ and} \\ & \exists \ell \text{ s.t. } i, j \in \mathcal{N}_\ell; \\ 1, & \text{if } (i, j) \in \mathcal{E} \text{ or } (j, i) \in \mathcal{E}, \text{ and} \\ & \exists \ell_1 \neq \ell_2 \text{ s.t. } i \in \mathcal{N}_{\ell_1}, j \in \mathcal{N}_{\ell_2}; \\ 0, & \text{otherwise.} \end{cases}$$

Clearly,  $\bar{S}$  defined in (1.2) is *sparse*, due to our assumption on the underlying community structure in  $\mathcal{G}$ , and  $\bar{L} = D - \bar{S}$  is *low-rank*. Indeed,  $\bar{L}$  is a BDO obtained by completing the clusters into cliques and deleting the inter-cluster edges; therefore,  $\bar{L}$  is low rank because each block has rank one and  $\text{rank}(\bar{L})$  is equal to the number of diagonal blocks, i.e.,  $\text{rank}(\bar{L}) = r$ .

In this paper, we propose a convex model similar to (1.1) of which optimal solution is equal to  $(\bar{L}, \bar{S})$ , defined as in (1.2), with very high probability. In particular, by adding constraints  $L \succeq 0$ ,  $\text{diag}(L) = \mathbf{1}$ ,  $L \geq 0$ , and  $|S_{ij}| \leq 1$  for  $1 \leq i \neq j \leq n$  to (1.1), we will obtain a tighter convex model. Another important property of our model is its ability to handle cases where  $D$  is partially observed. We will discuss this property in more detail in the next section, and develop an alternating direction method of multipliers (ADMM) algorithm for the proposed model. Finally, in Section 3, we first discuss how to generate a random family of networks for which modularity maximization fails, then compare ours with Louvain method [4], which is a greedy algorithm to solve the modularity maximization problem. While we were working on this idea independently, we found out a similar work by Chen et al. [8], which is also based on decomposition of the adjacency matrix by solving (1.1) when  $D$  is partially observed. In the next section, we will discuss the similarities among the two methods, and emphasize the advantages of our method over the one in [8]; and compare both methods in Section 3. Numerical results show that our model is indeed tighter than the robust PCA formulation, and is able to find the true clustering almost every time, while both Louvain method and the one in [8] fail when the variance among cluster sizes increases. The MATLAB code is available at <http://www2.ie.psu.edu/aybat/codes.html>.

## 2 Methodology

**2.1 Theoretical results** Model (1.1) was proposed by Candès et al. [7], and it is shown under some technical conditions on the components of  $D$  that its solution recovers the low rank and sparse components of the data matrix exactly with high probability - see [7] for more details. When  $D$  is *partially* observed, Tao and Yuan [21] proposed the following model:

$$(2.3) \quad \min_{L, S \in \mathbb{R}^{n \times n}} \{\|L\|_* + \rho \|S\|_1 : \pi_\Omega(L + S) = \pi_\Omega(D)\},$$

where  $\Omega \subset \{(i, j) : 1 \leq i, j \leq n\}$  is the set of observable indices and  $\pi_\Omega$  is the projection operator. For an arbitrary matrix  $Z$ ,  $(\pi_\Omega(Z))_{ij} = Z_{ij}$  when  $(i, j) \in \Omega$  and  $(\pi_\Omega(Z))_{ij} = 0$  otherwise. This model inspired us to develop a new method for network clustering based on convex optimization.

In order to compute a clustering for a partially observed network  $\mathcal{G}$ , Chen et al. [8] proposed to solve (2.3) repeatedly for different values of  $\rho$ , where  $D$  is the node-node incidence matrix of  $\mathcal{G}$  with a diagonal of ones. In particular, the authors of [8] proposed doing bisection on  $\rho$  until  $L_\rho^*$  is a BDO, which they called a *valid result*. The main advantage of the proposed method in this paper over the one in [8] is that we solve a *tighter* convex problem *one time* with the same complexity of solving (1.1), while in [8] the authors propose to solve (2.3) repeatedly for different values of  $\rho$ . Moreover, our numerical results show that our method is not only better in computation time, but also in clustering quality. Indeed, when the network size and/or the variance among cluster sizes increase, the method in [8] fails to cluster correctly while ours always succeeds.

**Assumption 1.** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{E})$  be an undirected network with  $\mathcal{N} = \{1, \dots, n\}$ , and  $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ . Suppose that  $\{\mathcal{N}_\ell\}_{\ell=1}^r$  be a partition of  $\mathcal{N}$  representing the non-overlapping communities in  $\mathcal{G}$ . Let  $\Omega := \bar{\Omega} \cup \mathcal{D}$  denote the set of observable entries of the adjacency matrix  $D$  such that  $\mathbf{diag}(D) = \mathbf{1}$ , where  $\mathcal{D} = \{(i, i) : i \in \mathcal{N}\}$ , and  $\bar{\Omega} \subset \mathcal{N} \times \mathcal{N} \setminus \mathcal{D}$ . Assume that an edge exists between two nodes in the same cluster with probability  $p$ , and it exists between two nodes from two different clusters with probability  $q$  such that  $p \gg q$ ; and for any two nodes whether there is an edge between them or not is known with probability  $p_0$ .

In this paper, we propose an efficient method that can recover the underlying community structure of  $\mathcal{G}$  by decomposing  $D$  into  $(\bar{L}, \bar{S})$  defined as in (1.2) with very high probability under Assumption 1. In such decomposition, the community structure is encoded in  $\bar{L}$ , and the off-diagonal non-zero entries of  $\bar{S}$  correspond to the node pairs that are in the same cluster but not connected by any edge, or to the edges connecting two different clusters. Following [8], the total number of off-diagonal non-zeros in  $\bar{S}$ , denoted by  $\|\bar{S}\|_0$ , will be called total number of *disagreements*. Before we introduce our model, we investigate some properties of  $(\bar{L}, \bar{S})$ .

**LEMMA 2.1.** Given  $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ , and  $\Omega \subset \mathcal{N} \times \mathcal{N}$ , define  $\chi := \{(L, S) \in \mathbb{S}_n \times \mathbb{S}_n : \pi_\Omega(L + S) = \pi_\Omega(D), |S_{ij}| \leq 1 \ \forall 1 \leq i \neq j \leq n, \mathbf{diag}(L) = \mathbf{1}, L \geq \mathbf{0}_n, L \succeq \mathbf{0}_n\}$ , where  $\mathbb{S}_n$  denotes the subspace of  $n \times n$  symmetric real matrices. Under Assumption 1, we have  $(\bar{L}, \bar{S}) \in \chi$ .

*Proof.* Directly from the definition of  $\bar{S}$  in (1.2), and from the facts:  $\bar{L} := D - \bar{S}$ ,  $\mathbf{diag}(D) = \mathbf{1}$ , it follows that the first four constraints are satisfied at  $(\bar{L}, \bar{S})$ . Therefore it is enough to show that  $\bar{L}$  is positive semidefinite. Under Assumption 1,  $\bar{L}$  is BDO with

$r$  blocks since  $\{\mathcal{N}_\ell\}_{\ell=1}^r$  is a partition of  $\mathcal{N}$ . Now let  $v^\ell \in \mathbb{R}^n$  be such that  $v_j^\ell = 1$  if  $j \in \mathcal{N}_\ell$ , and  $v_j^\ell = 0$  otherwise. It is easy to show that  $\bar{L} = \sum_{\ell=1}^r v^\ell (v^\ell)^T$ . Therefore,  $\bar{L} \succeq \mathbf{0}_n$  such that  $\mathbf{rank} \bar{L} = r$ .

Based on Lemma 2.1, we define our new model as:

$$\begin{aligned} (L^*, S^*) \in \operatorname{argmin}_{L, S \in \mathbb{S}_n} & \mathbf{Tr}(L) + \rho \|S\|_1 \\ \text{s.t.} & \pi_\Omega(L + S) = \pi_\Omega(D), \\ & \mathbf{diag}(S) = \mathbf{0}, |S_{ij}| \leq 1 \ \forall i \neq j, \\ & L \succeq \mathbf{0}_n, L \geq \mathbf{0}_n. \end{aligned} \quad (2.4)$$

Note that in (2.4), we replaced  $\|L\|_*$  in (2.3) with  $\mathbf{Tr}(L)$ , and also replaced  $\mathbf{diag}(L) = \mathbf{1}$  constraint in the definition of  $\chi$  with  $\mathbf{diag}(S) = \mathbf{0}$  constraint. The first follows from the fact that  $L \succeq \mathbf{0}_n$  implies  $\|L\|_* = \mathbf{Tr}(L)$ . Indeed, for a positive semidefinite matrix  $L$ , its non-zero singular values  $\{\sigma_i\}_{i=1}^{\mathbf{rank}(L)}$  are equal to its non-zero eigenvalues  $\{\lambda_i\}_{i=1}^{\mathbf{rank}(L)}$ . Therefore, we have  $\|L\|_* = \sum_{i=1}^n \sigma_i = \sum_{i=1}^n \lambda_i = \mathbf{Tr}(L)$ . Moreover, since  $\mathcal{D} \subset \Omega$  and  $\mathbf{diag}(D) = \mathbf{1}$ ,  $\mathbf{diag}(L) = \mathbf{1}$  if and only if  $\mathbf{diag}(S) = \mathbf{0}$ . Note that replacing  $\mathbf{diag}(L) = \mathbf{1}$  equivalently with  $\mathbf{diag}(S) = \mathbf{0}$  is the key point for developing an ADMM algorithm with efficiently solvable subproblems, which will be discussed in the next section.

The following two theorems show the importance and special properties of our formulation. Theorem 2.1 and Theorem 2.2 were originally proved in [8] for model (2.3). Since the feasible region of our model in (2.4) is a subset of that in (2.3), these important results trivially extend to our formulation as well.

**THEOREM 2.1.** For any  $\rho > 0$ , if  $L^*$  in (2.4) is a BDO, then it provides the optimal clustering in the sense that the total number of observed disagreements, i.e.,  $\|\pi_\Omega(S^*)\|_0$ , is minimized.

*Proof.* See proof of Theorem 2 in [8].

**THEOREM 2.2.** Given  $\mathcal{G} = (\mathcal{N}, \mathcal{E})$  and  $\Omega \subset \mathcal{N} \times \mathcal{N}$  satisfying Assumption 1. Let  $\{\mathcal{N}_\ell\}_{\ell=1}^r$  represent the true underlying community structure of  $\mathcal{G}$ , and  $\bar{L} = D - \bar{S}$ , where  $\bar{S}$  is defined in (1.2). Then for all  $c > 0$ , there exists  $C > 0$  such that with probability of at least  $1 - cn^{-10}$ ,  $(\bar{L}, \bar{S})$  is the unique optimal solution of (2.4) when  $\rho = \frac{1}{32\sqrt{np_0}}$ , provided that

$$(2.5) \quad n \log^2 n \leq CK_{\min}^2 p_0 (1 - 2\gamma)^2,$$

where  $\gamma = \max\{1 - p, q\}$  and  $K_{\min} := \min\{|\mathcal{N}_\ell| : 1 \leq \ell \leq r\}$  is the size of the smallest cluster.

*Proof.* Given  $c > 0$ , Theorem 2 in [8] shows that there exists  $C > 0$  such that  $(\bar{L}, \bar{S})$  is the unique optimal

solution to (2.3) with probability at least  $1 - cn^{-10}$  provided that (2.5) holds. Lemma 2.1 implies that  $(\bar{L}, \bar{S})$  is feasible to (2.4); hence, it must be an optimal solution to the more tighter problem in (2.4) as well. Moreover, under the assumptions of Theorem 2.2, (2.3) has a unique solution with high probability, which implies that  $(\bar{L}, \bar{S})$  is the unique optimal solution to (2.4) w.p. at least  $1 - cn^{-10}$ .

**2.2 Algorithm** In this section, we develop an ADMM algorithm to solve (2.4). Define  $\phi \subset \mathbb{S}_n \times \mathbb{S}_n$  as

$$(2.6) \quad \phi := \left\{ (X, S) : \begin{array}{l} \pi_\Omega(X + S) = \pi_\Omega(D), \\ X \succeq \mathbf{0}_n, \text{diag}(S) = \mathbf{0}, \\ |S_{ij}| \leq 1, \quad 1 \leq i \neq j \leq n \end{array} \right\}.$$

By using partial variable splitting as in [1, 2], (2.4) can be written equivalently as follows:

$$(2.7) \quad \begin{aligned} (L^*, L^*, S^*) \in \underset{L, X, S \in \mathbb{S}_n}{\operatorname{argmin}} \quad & \operatorname{Tr}(L) + \rho \|S\|_1 \\ \text{s.t.} \quad & X = L, \quad L \succeq \mathbf{0}_n, \quad (X, S) \in \phi. \end{aligned}$$

Let  $\mathbb{S}_n^+$  denote the cone of  $n \times n$  symmetric positive semidefinite matrices. Given a penalty parameter  $\mu > 0$ , the partial augmented Lagrangian [3] of (2.7) is defined for any  $L \in \mathbb{S}_n^+$ ,  $(X, S) \in \phi$ , and  $Y \in \mathbb{S}_n$  as follows

$$\begin{aligned} \mathcal{L}_\mu(L, X, S; Y) = \\ \operatorname{Tr}(L) + \rho \|S\|_1 + \langle Y, X - L \rangle + \frac{\mu}{2} \|X - L\|_F^2. \end{aligned}$$

Given  $Y \in \mathbb{S}_n$ , since it is not easy to minimize  $\mathcal{L}_\mu(L, X, S; Y)$  jointly in  $(L, X, S) \in \mathbb{S}_n^+ \times \phi$ , the *method of multipliers* is not a practical approach to solve (2.7). On the other hand, given  $Y$ , alternating minimization of  $\mathcal{L}_\mu(L, X, S; Y)$  in  $(X, S) \in \phi$  for fixed  $L$ , and in  $L \in \mathbb{S}_n^+$  for fixed  $(X, S)$  can be done efficiently. Therefore, we propose ADMIPC, which is an ADMM algorithm with increasing penalty sequence, to solve (2.7). Each step of ADMIPC is displayed in Figure 1. The subproblems in Step 6 and Step 5 are the computational bottlenecks, and they can be solved efficiently as explained in Lemma 2.2 and Lemma 2.3. The initialization part will be discussed in Section 3.2.

The convergence of ADMIPC directly follows from [14]. Indeed, the variable penalty ADMM algorithms in [12, 13, 14] are proposed to solve variational inequalities (VI) of the form:

$$\begin{aligned} (x - x^*)^\top F(x^*) + (y - y^*)^\top G(y^*) \geq 0, \quad \forall (x, y) \in \Omega \\ \Omega := \{(x, y) : x \in \mathcal{X}, y \in \mathcal{Y}, Ax + By = b\}, \end{aligned}$$

where  $A \in \mathbb{R}^{m \times n_1}$ ,  $B \in \mathbb{R}^{m \times n_2}$ , and  $b \in \mathbb{R}^m$ . The convergence proofs in [12, 13, 14] require that both  $F : \mathcal{X} \rightarrow \mathbb{R}^{n_1}$  and  $G : \mathcal{Y} \rightarrow \mathbb{R}^{n_2}$  be *continuous point-to-point*

*maps* that are monotone with respect to the non-empty closed convex sets  $\mathcal{X} \subset \mathbb{R}^{n_1}$  and  $\mathcal{Y} \subset \mathbb{R}^{n_2}$ , respectively. When these variable penalty ADMM methods for VI are applied to the VI reformulation of convex optimization problems of the form  $\min\{f(x) + g(y) : (x, y) \in \Omega\}$ , the requirement that  $F$  and  $G$  be continuous point-to-point maps implies that  $F(x) = \nabla f(x)$ , and  $G(y) = \nabla g(y)$ . On the other hand, when  $f$  (similarly  $g$ ) is a non-smooth convex function,  $F$  (similarly  $G$ ) is the subdifferential operator, which is a *point-to-set map*; therefore, the convergence proofs for variable penalty ADMM algorithms in [12, 13, 14] do not extend to non-smooth convex optimization problems – see Assumption A and the following discussion on page 107 in [14]. However, even though the objective in (2.7) is non-smooth, the following result establishes that the convergence of ADMIPC follows from [14].

**THEOREM 2.3.** *Let  $Z_k = (L_k, X_k, S_k, Y_k)$  denote the iterates generated by ADMIPC in Figure 1, and  $\mathcal{Z}^*$  denote the set of optimal primal-dual pairs to (2.7), i.e.,  $(L^*, X^*, S^*, Y^*) \in \mathcal{Z}^*$  if and only if*

$$\begin{aligned} \langle \mathbf{I}_n - Y^*, L - L^* \rangle &\geq 0, \quad \forall L \succeq \mathbf{0}_n, \\ \rho \langle G, S - S^* \rangle + \langle Y^*, X - X^* \rangle &\geq 0, \quad \forall (X, S) \in \phi, \\ X^* &= L^*, \quad G \in \partial \|S^*\|_1. \end{aligned}$$

*Then  $\min\{\|Z_k - Z\|_F : Z \in \mathcal{Z}^*\} \rightarrow 0$ . Moreover,  $\{Z_k\}$  is bounded.*

*Proof.* Using the change of variables  $S := S^+ - S^-$  for  $S^+, S^- \succeq \mathbf{0}_n$ ,  $\|S\|_1$  can be equivalently written as  $\langle \mathbf{E}_n, S^+ + S^- \rangle$ , where  $\mathbf{E}_n \in \mathbb{R}^{n \times n}$  is a matrix of ones. Consider

$$(2.8) \quad \min \left\{ \operatorname{Tr}(L) + \rho \langle \mathbf{E}_n, S^+ + S^- \rangle : \begin{array}{l} X = L, \quad L \succeq \mathbf{0}_n, \\ (X, S^+, S^-) \in \phi' \end{array} \right\},$$

where  $\phi' \subset \prod_{i=1}^3 \mathbb{S}_n$  is defined as

$$\phi' := \left\{ (X, S^+, S^-) : \begin{array}{l} \pi_\Omega(X + S^+ - S^-) = \pi_\Omega(D), \\ X \succeq \mathbf{0}_n, \quad S^+ \succeq \mathbf{0}_n, \quad S^- \succeq \mathbf{0}_n, \\ \text{diag}(S^+) = \text{diag}(S^-) = \mathbf{0}, \\ S_{ij}^+ + S_{ij}^- \leq 1, \quad \forall i \neq j \end{array} \right\}.$$

Note that (2.8) is a smooth convex optimization problem equivalent to (2.7), and it satisfies all the assumption in [14]. Given a nondecreasing penalty sequence  $\{\mu_k\}$  such that  $\sup_k \mu_k < \infty$ , let  $\{(\tilde{L}_k, \tilde{X}_k, \tilde{S}_k^+, \tilde{S}_k^-, \tilde{Y}_k)\}$  be the iterate sequence generated by the variable penalty ADMM in [14] when the augmented Lagrangian of (2.8) is minimized alternatingly in  $(X, S^+, S^-) \in \phi'$ , and in  $L \succeq \mathbf{0}_n$ . Define  $\tilde{Z}_k := (\tilde{L}_k, \tilde{X}_k, \tilde{S}_k, \tilde{Y}_k)$ , where

$\tilde{S}_k := \tilde{S}_k^+ - \tilde{S}_k^-$ . It is easy to see that  $\{\tilde{Z}_k\}$  would be the same with the one generated by ADMIPC in Figure 1, i.e.,  $\tilde{Z}_k = Z_k$  for all  $k \geq 1$ . The result follows from Theorem 4 in [14], which shows that  $\{\tilde{Z}_k\}$  is bounded and  $\min\{\|\tilde{Z}_k - Z\|_F : Z \in \mathcal{Z}^*\} \rightarrow 0$ .

---

**Algorithm ADMIPC** ( $\rho, \{\mu_k\}_{k \in \mathbb{Z}_+}$ )

---

- 1: **Input:**  $\rho > 0, \{\mu_k\}_{k \in \mathbb{Z}_+} \subset \mathbb{R}_{++}$  s.t.  $\mu_{k+1} \geq \mu_k$ , and  $\sup_k \mu_k < \infty$
  - 2: **Initialization:**  $k = 0; L_0 = \mathbf{0}_n$ ;
  - 3:  $Y_0 = \frac{\pi_\Omega(D)}{\max\{\|\pi_\Omega(D)\|_2, \rho^{-1}\|\pi_\Omega(D)\|_\infty\}}$ ;
  - 4: **while** not converged **do**
  - 5:  $(X_{k+1}, S_{k+1}) \leftarrow \operatorname{argmin}\{\rho\|S\|_1 + \frac{\mu_k}{2}\|X - L_k + \frac{Y_k}{\mu_k}\|_F^2 : (X, S) \in \phi\}$
  - 6:  $L_{k+1} \leftarrow \operatorname{argmin}\{\operatorname{Tr}(L) + \frac{\mu_k}{2}\|L - X_{k+1} - \frac{Y_k}{\mu_k}\|_F^2 : L \succeq 0\}$
  - 7:  $Y_{k+1} \leftarrow Y_k + \mu_k(X_{k+1} - L_{k+1})$
  - 8:  $k \leftarrow k + 1$
  - 9: **end while**
- 

Figure 1: ADMIPC: Alternating Direction Method with Increasing Penalty for Clustering

Lemma 2.2 shows that the subproblem in Step 6 can be solved efficiently by computing a partial-eigenvalue decomposition of an  $n \times n$  matrix.

LEMMA 2.2. *The solution to the subproblem in Step 6 can be written in closed form:*

$$(2.9) \quad L_{k+1} = W \operatorname{diag}(\max\{\lambda_k - \mu_k^{-1}\mathbf{1}, \mathbf{0}\}) W^T,$$

where  $W \operatorname{diag}(\lambda_k) W^T$  is the eigenvalue decomposition of  $Q_k^X := X_{k+1} + \frac{Y_k}{\mu_k}$ .

*Proof.* Since  $\operatorname{Tr}(L) = \langle \mathbf{I}_n, L \rangle$ , the subproblem in Step 6 can be equivalently written as

$$(2.10) \quad L_{k+1} = \operatorname{argmin}_{L \succeq 0} \|L - (Q_k^X - \mu_k^{-1}\mathbf{I}_n)\|_F.$$

Since  $Q_k^X - \mu_k^{-1}\mathbf{I}_n = W \operatorname{diag}(\lambda_k - \mu_k^{-1}\mathbf{1}) W^T$ , (2.9) follows from the properties of Euclidean projection onto the positive semidefinite cone of symmetric matrices.

One of the main reasons of using an increasing sequence of penalties  $\{\mu_k\}$  in ADMIPC is because the work required for eigenvalue decomposition in Step 6 reduces significantly as fewer leading eigenvalues are needed for small values of  $\mu_k$ . Note that according to (2.9), we do not need to compute eigenvalues of  $Q_k^X$  that are smaller than  $\mu_k^{-1}$ . Indeed,  $Q_k^X$  may not be low rank, and many of its eigenvalues may be large during the initial iterations in the transient phase of the algorithm. This could make Step 6 an expensive operation for a constant penalty

ADMM method with  $\mu_k = \mu$ , as there may be many leading eigenvalues that are larger than  $\mu$ . However, based on (2.9), by choosing small values for  $\mu_k$  during the initial iterations and then gradually increasing it, we can avoid computing all the eigenvalues of  $Q_k^X$ . Refer to [1] for more details about this concept. Moreover, it is shown in [14] that results of Theorem 2.3 are still true if Step 6 is computed inexactly. Since with very high probability the optimal solution  $L^*$  is unique and equal to  $\bar{L}$ , which has low rank, Step 6 can be computed approximately by calculating only a small number of leading eigenvalues, and the approximation error will be small for all sufficiently large  $k$ . Indeed, Theorem 2.3 implies that with high probability  $X_k \rightarrow L^* = \bar{L}$  (due to uniqueness of  $L^*$ ). Hence, for any  $\delta > 0$ , there exists  $\{\mu_k\}$  such that  $\|Q_k^X - X_{k+1}\|_2 \leq \frac{\delta}{2}$  due to boundedness of  $\{Y_k\}$ , and  $\|X_{k+1} - \bar{L}\|_2 \leq \frac{\delta}{2}$ , where  $\|\cdot\|_2$  denotes the spectral norm. Thus,  $\|Q_k^X - \bar{L}\|_2 \leq \delta$ . Moreover, since eigenvalues of a matrix is a continuous function of its entries, it follows that for all  $\delta > 0$ , there exists  $\{\mu_k\}$  such that  $\|\lambda_k - \bar{\lambda}\|_\infty \leq \delta$ , where  $\bar{\lambda} \in \mathbb{R}^n$  denotes the vector of eigenvalues of  $\bar{L}$ . Since the number of nonzeros in  $\bar{\lambda}$  is  $r \ll n$ ,  $n - r$  components of  $\lambda_k$  is between  $-\delta$  and  $\delta$ .

In order to compute the eigenvalue decomposition of  $Q_k^X$  in Step 6, we used LANSVD routine in PROPACK package. LANSVD routine is based on the Lanczos bidiagonalization algorithm with partial reorthogonalization for computing partial singular value decomposition (SVD). Let  $\tilde{\lambda} := \lambda_k - \mu_k^{-1}\mathbf{1}$ , and  $U \operatorname{diag}(\sigma) V^T$  denote SVD of  $Q_k^X - \mu_k^{-1}\mathbf{I}_n$ , where  $U_i, V_i$  denote left and right singular vectors corresponding to  $i$ -th singular value  $\sigma_i$ . It is clear that if  $U_i^T V_i = 1$ , then  $\tilde{\lambda}_i = \sigma_i > 0$ ; and if  $U_i^T V_i = -1$ , then  $\tilde{\lambda}_i = -\sigma_i < 0$ . Hence, (2.9) can be computed efficiently using a partial SVD.

LEMMA 2.3. *Let  $\Omega := \bar{\Omega} \cup \mathcal{D}$  denote the set of observable entries of the adjacency matrix  $D$  such that  $\operatorname{diag}(D) = \mathbf{1}$ , where  $\mathcal{D} = \{(i, i) : i \in \mathcal{N}\}$ , and  $\bar{\Omega} \subset \mathcal{N} \times \mathcal{N} \setminus \mathcal{D}$ . The solution to the subproblem in Step 5 can be written in closed form:*

$$\begin{aligned} C_1 &= \operatorname{sgn}(\pi_{\bar{\Omega}}(D - Q_k^L)), \\ C_2 &= \max\{|\pi_{\bar{\Omega}}(D - Q_k^L)| - \rho\mu_k^{-1}\mathbf{E}_n, \mathbf{0}_n\}, \\ S_{k+1} &= \min\{\pi_{\bar{\Omega}}(D), \max\{-\mathbf{E}_n, C_1 \odot C_2\}\}, \\ X_{k+1} &= \pi_\Omega(D - S_{k+1}) + \max\{\pi_{\Omega^c}(Q_k^L), \mathbf{0}_n\}, \end{aligned}$$

where  $Q_k^L := L_k - \frac{Y_k}{\mu_k}$ ,  $\mathbf{E}_n \in \mathbb{R}^{n \times n}$  is a matrix of ones, and  $\odot$  represents the component-wise multiplication.

*Proof.* The subproblem in step 5 can be written as

$$(2.11) \quad (X_{k+1}, S_{k+1}) = \operatorname{argmin}_{(X, S) \in \phi} \rho\|S\|_1 + \frac{\mu_k}{2}\|X - Q_k^L\|_F^2.$$

For  $(X, S) \in \phi$ , we have

$$(2.12) \quad \begin{aligned} X - Q_k^L &= \pi_{\bar{\Omega}}(D - S - Q_k^L) \\ &+ \pi_{\mathcal{D}}(D - Q_k^L) + \pi_{\Omega^c}(X - Q_k^L). \end{aligned}$$

Moreover, from the optimality conditions for (2.11), it is clear that  $(S_{k+1})_{ij} = 0$  for all  $(i, j) \in \mathcal{D} \cup \Omega^c$ . Therefore, (2.11) is equivalent to the following problem:

$$(2.13) \quad \begin{aligned} \min_{(X, S) \in \mathbb{S}_n \times \mathbb{S}_n} \quad & \rho \|\pi_{\bar{\Omega}}(S)\|_1 + h(X, S) \\ \text{s.t.} \quad & \mathbf{0}_n \leq \pi_{\Omega^c}(X), \\ & -\pi_{\bar{\Omega}}(\mathbf{E}_n) \leq \pi_{\bar{\Omega}}(S) \leq \pi_{\bar{\Omega}}(D), \end{aligned}$$

where  $h(X, S) := \frac{\mu_k}{2} \|\pi_{\bar{\Omega}}(S) - \pi_{\bar{\Omega}}(D - Q_k^L) - \pi_{\Omega^c}(X - Q_k^L)\|_F^2$ . For the sake of notational simplicity, let  $\tilde{S}_{ij} := (D - Q_k^L)_{ij}$  for all  $(i, j) \in \bar{\Omega}$ . Note that (2.13) is separable over  $(i, j)$ . Therefore, for all  $(i, j) \in \bar{\Omega}$ ,

$$(2.14) \quad \begin{aligned} (S_{k+1})_{ij} &= \operatorname{argmin}_{S_{ij} \in \mathbb{R}} \rho |S_{ij}| + \frac{\mu_k}{2} (S_{ij} - \tilde{S}_{ij})^2 \\ \text{s.t.} \quad & -1 \leq S_{ij} \leq D_{ij}. \end{aligned}$$

Given  $\bar{t} \in \mathbb{R}$  and  $\mu > 0$ , define  $f : \mathbb{R} \rightarrow \mathbb{R}$  such that  $f(t) = \rho |t| + \frac{\mu}{2} (t - \bar{t})^2$ . Let  $t_u^* := \operatorname{argmin}_{t \in \mathbb{R}} f(t)$  and  $t_c^* = \operatorname{argmin}_{t \in \mathbb{R}} \{f(t) : a \leq t \leq b\}$ . Since  $f$  is convex on  $\mathbb{R}$ , it is easy to show that

$$(2.15) \quad \begin{aligned} t_u^* &= \operatorname{sgn}(\bar{t}) \max \left\{ |\bar{t}| - \frac{\rho}{\mu}, 0 \right\}, \\ t_c^* &= \min \{b, \max\{a, t_u^*\}\}. \end{aligned}$$

Thus, (2.15) implies that for all  $(i, j) \in \bar{\Omega}$ , we have

$$(2.16) \quad \begin{aligned} (S_{k+1})_{ij} &= \min \{D_{ij}, \max \{-1, c_{ij}\}\}, \\ c_{ij} &= \operatorname{sgn}(\tilde{S}_{ij}) \max \left\{ \left| \tilde{S}_{ij} \right| - \frac{\rho}{\mu_k}, 0 \right\}. \end{aligned}$$

The structure of  $S_{k+1}$  follows from (2.16) and the fact that  $(S_{k+1})_{ij} = 0$  for all  $(i, j) \in \mathcal{D} \cup \Omega^c$ .

Since  $(X_{k+1}, S_{k+1}) \in \phi$ , clearly for all  $(i, j) \in \Omega$ , we have  $(X_{k+1})_{ij} = (D - S_{k+1})_{ij}$ . Moreover, (2.12) implies that for all  $(i, j) \in \Omega^c$ , we have

$$(2.17) \quad \begin{aligned} (X_{k+1})_{ij} &= \operatorname{argmin}_{X_{ij} \geq 0} (X_{ij} - (Q_k^L)_{ij})^2, \\ &= \max \{(Q_k^L)_{ij}, 0\}. \end{aligned}$$

### 3 Numerical results

In Section 3.3.1, we compared our formulation (2.4) with the robust PCA formulation (2.3), which is adopted by Chen et al. in [8]. Numerical results show that our formulation is more tighter, and is able to recover many clusters which cannot be detected using the methodology given in [8]. Next, in Section 3.3.2, we compared

ADMIPC with Louvain method, which is based on modularity maximization, on randomly generated test problems. The results show that as the number of nodes in the network increases, Louvain method starts merging small clusters; and this phenomena becomes more apparent when the variation among cluster sizes increases. The empirical results presented in Section 3.3.2 indeed confirm that resolution limit [10, 15] becomes a major drawback for modularity maximization.

**3.1 Random network generation.** In this section we describe the random network generation used in our experiments. Let  $\mathcal{G} = (\mathcal{N}, \mathcal{E})$  be a random undirected network, and  $\{\mathcal{N}_\ell\}_{\ell=1}^r$ , a partition of  $\mathcal{N}$ , be the underlying clustering in  $\mathcal{G}$  chosen such that

$$\mathcal{N}_\ell := \left\{ \sum_{i=1}^{\ell-1} n_i + 1, \dots, \sum_{i=1}^{\ell} n_i \right\}, \quad \forall \ell \in \{1, \dots, r\},$$

where  $n_\ell := |\mathcal{N}_\ell|$  denote the size of  $\ell$ -th cluster. Let  $0 < \alpha \leq 1$  be a parameter that will control the variation among cluster sizes  $\{n_\ell\}_{\ell=1}^r$ . Note that  $n = \sum_{\ell=1}^r \frac{1-\alpha}{1-\alpha^r} n \alpha^{\ell-1}$ . Given  $x > 0$ , let  $[x]$  denote the nearest integer to  $x$ . The cluster sizes are chosen as

$$(3.18) \quad n_\ell = \left\lceil \frac{1-\alpha}{1-\alpha^r} n \alpha^{\ell-1} \right\rceil, \quad \forall \ell \in \{1, \dots, r\}.$$

In our experiments, we choose  $r = \lceil 0.05n \rceil$ . For instance, suppose  $n = 100$ , then the total number of clusters is  $r = 5$ ; Table 3.1 displays the size of each cluster for different values of  $\alpha$ . For the sake of simplicity, assume that  $n = \sum_{\ell=1}^r n_\ell$ , and  $n_\ell > 0$  for all  $\ell = 1, \dots, r$ ; otherwise, we define  $\mathcal{N}_\ell$  only for  $\ell$  such that  $n_\ell > 0$ , and reset  $r = |\{\ell : n_\ell > 0\}|$ .

$\alpha$	$n_1$	$n_2$	$n_3$	$n_4$	$n_5$
<b>1</b>	20	20	20	20	20
<b>0.9</b>	24	22	20	18	16
<b>0.8</b>	30	24	19	15	12
<b>0.7</b>	36	25	18	12	9
<b>0.6</b>	43	26	16	9	6
<b>0.5</b>	52	26	13	6	3

Table 1: Cluster sizes for different values of  $\alpha$  when  $n = 100$  and  $r = 5$ .

In the next step, after we choose the underlying clustering  $\{\mathcal{N}_\ell\}_{\ell=1}^r$  as above, we generated the edges in  $\mathcal{G}$  as follows. Let  $\mathcal{U} = \{(i, j) \in \mathcal{N} \times \mathcal{N} : i < j\}$ , and  $\mathcal{E}^U \subset \mathcal{U}$  be such that  $|\mathcal{E}^U| = \lceil 0.05|\mathcal{U}| \rceil$  elements are randomly chosen with equal probability; define  $\mathcal{E}^L := \{(i, j) : (j, i) \in \mathcal{E}^U\}$ , and  $\bar{\mathcal{E}} := \{(i, j) \in \mathcal{N} \times \mathcal{N} : \exists \ell \in \{1, \dots, r\} \text{ s.t. } i \in \mathcal{N}_\ell, j \in \mathcal{N}_\ell\}$ . Then we set  $\mathcal{E}$  as the symmetric difference of  $\bar{\mathcal{E}}$  and  $\mathcal{E}^U \cup \mathcal{E}^L$ , i.e.,  $\mathcal{E} := \bar{\mathcal{E}} \Delta (\mathcal{E}^U \cup \mathcal{E}^L)$ . Note that  $\mathcal{G} = (\mathcal{N}, \mathcal{E})$  generated this way satisfies Assumption 1.

Let  $D \in \mathbb{S}_n$  be the node-node incidence matrix corresponding to  $\mathcal{G}$  such that  $\text{diag}(D) = \mathbf{1}$ , i.e.,  $D_{ij} = 1$  if  $(i, j) \in \mathcal{E}$  or  $i = j$ , and  $D_{ij} = 0$  otherwise. Let  $\Omega \subset \mathcal{N} \times \mathcal{N}$  be the set of indices corresponding to the observable entries of  $D$ . Note that according to Assumption 1, for any given  $i \in \mathcal{N}$  and  $j \in \mathcal{N}$ , whether  $i$  and  $j$  are connected by an edge in  $\mathcal{E}$  or not is known with probability  $p_0$ . Hence, to generate  $\Omega$ , let  $\Omega^U$  be the set of  $N_o := \left\lceil \frac{p_0(n^2-n)}{2} \right\rceil$  indices of the upper triangular entries of  $D$  chosen uniformly at random, i.e.,  $\Omega^U \subset \mathcal{U}$  such that  $|\Omega^U| = N_o$ . Since  $D$  is symmetric, the symmetric lower triangular elements  $\Omega^L := \{(i, j) : (j, i) \in \Omega^U\}$  should be in  $\Omega$ ; and it's also known that the diagonal entries are all ones. Therefore, we set  $\Omega := \Omega^U \cup \Omega^L \cup \mathcal{D}$ .

**3.2 Initialization and stopping criterion.** In all the experiments, we set  $\rho = \frac{1}{\sqrt{n}}$ ,  $L_0 = \mathbf{0}_n$ ,  $\mu_0 = \frac{1.25}{\|\pi_\Omega(D)\|_2}$ , and  $Y_0 = \frac{\pi_\Omega(D)}{\max\{\|\pi_\Omega(D)\|_2, \rho^{-1}\|\pi_\Omega(D)\|_\infty\}}$  in ADMIPC. The penalty multiplier sequence  $\{\mu_k\}$  is chosen such that  $\mu_{k+1} = \min\{\kappa\mu_k, \bar{\mu}\}$  for  $k \geq 1$ , where  $\bar{\mu} = 10^7$  and  $\kappa = 1.2$ . We terminate the algorithm when the following primal-dual stopping conditions hold:

$$(3.19) \quad \|L_{k+1} - X_{k+1}\|_F \leq \text{tol}_p, \quad \frac{\mu_k \|L_{k+1} - L_k\|_F}{\|\pi_\Omega(D)\|_F} \leq \text{tol}_d,$$

where  $\text{tol}_p = \epsilon_r \max\{\|L_{k+1}\|_F, \|X_{k+1}\|_F\}$ ,  $\text{tol}_d = \epsilon_r \|Y_{k+1}\|_F$ , and  $\epsilon_r = 5 \times 10^{-4}$ . The first equation gives the primal stopping criterion and the second one gives the dual stopping criterion. For more details about the stopping criteria refer to [5].

**3.3 Results** All the numerical experiments were conducted on a Windows 7 machine with Intel Core i7-3520M Processor (4 MB cash, 2 cores at 2.9 GHz), and 16 GB RAM running MATLAB 8.2 (64 bit). We consider two different cases. In the first case, we assume that  $\mathcal{E}$  is perfectly known, i.e., all the entries of  $D$  are observed. In the second case, we assume that  $\mathcal{E}$  is partially observable, i.e., we only know the entries of  $D$  corresponding to indices in  $\Omega$ . For both cases, we compared ADMIPC with the method proposed in [8] and with Louvain method for different values of  $(n, \alpha)$ .

**3.3.1 ADMIPC vs ADMM on RPCA** First, for  $\rho = \frac{1}{\sqrt{n}}$ , we compare our formulation, given in (2.4), with the robust PCA (RPCA) formulation in (2.3) to check whether the proposed formulation (2.4) is tighter than (2.3). In particular, given randomly generated networks as described in Section 3.1, we solve (2.4) using

ADMIPC and compare the results with those obtained by solving (2.3) using a modified version of IALM in [16]. IALM is nothing but an increasing penalty ADMM method customized for (1.1). Note that IALM [16] works when  $D$  is fully observed, and it does not work on (2.3). However, Theorem 1.1 in [2] shows that (2.3) is equivalent to

$$(3.20) \quad \min_{L, S \in \mathbb{R}^{n \times n}} \{\|L\|_* + \rho \|\pi_\Omega(S)\|_1 : L + S = \pi_\Omega(D)\};$$

and one can easily modify IALM [16] to solve (3.20). We call the modified version as M-IALM( $\rho$ ). The results presented in this section show that, for  $\rho = 1/\sqrt{n}$ , our formulation (2.4) is indeed tighter than the RPCA formulation (2.3).

Next, we compared ADMIPC with the method developed in [8], which is based on RPCA formulation (2.3). We call the method in [8] as RPCA with bisection (RPCAB). RPCAB calls M-IALM on (2.3) for changing values of  $\rho$ . In particular, for a given  $\rho > 0$ , RPCAB calls M-IALM to compute  $L_\rho^*$ , the optimal low-rank component to (2.3). Next, if  $\text{Tr}(L_\rho^*) \neq n$ , then RPCAB updates  $\rho$  as follows: when  $\text{Tr}(L_\rho^*) > n$ ,  $\rho \leftarrow \rho/2$ ; otherwise,  $\rho \leftarrow 2\rho$ . After  $\rho$  is updated, RPCAB calls M-IALM on (2.3) with the new  $\rho$  value. In all the numerical tests we set the initial value of  $\rho = \frac{1}{\sqrt{n}}$ . Based on the discussion in [5] on stopping criteria for ADMM, the dual stopping criterion for M-IALM is chosen as in (3.19) such that  $\text{tol}_d = \epsilon_r \|Y_{k+1}\|_F$ ; and the primal stopping criterion for M-IALM is chosen as  $\|\pi_\Omega(D) - (L_{k+1} + S_{k+1})\|_F \leq \text{tol}_p$ , where  $\text{tol}_p = \epsilon_r \max\{\|L_{k+1}\|_F, \|S_{k+1}\|_F, \|\pi_\Omega(D)\|_F\}$ , and  $\epsilon_r = 5 \times 10^{-4}$ . The stopping criterion for RPCAB is set as  $|\text{Tr}(L_\rho^*) - n|/n \leq 0.01$ .

The following two cases are considered when we compare the low-rank component output by ADMIPC with those generated by M-IALM( $1/\sqrt{n}$ ) and by RPCAB. For each  $n \in \{100, 200, 300, 400, 500\}$ , random networks are generated as described in Section 3.1 for  $\alpha \in \{0.6, 0.7, \dots, 1\}$ . For each  $(n, \alpha)$  setting, we generated 10 random graphs, and corresponding  $D$ . In **Case 1**, all the entries of  $D$  are observed, i.e.,  $p_0 = 1$ , and in **Case 2**, we assume that  $\mathcal{E}$  is partially observable; hence, there are unobserved entries in  $D$ , i.e.,  $p_0 < 1$ .

Let  $\bar{L}$  represent the underlying clustering in  $\mathcal{G}$ , i.e.,  $\bar{L} = D - \bar{S}$  for  $\bar{S}$  defined in (1.2), and  $L^*$  is the optimal low-rank component computed by one of the algorithms mentioned above. By definition,  $\bar{L}$  is BDO with  $r$  diagonal blocks, each of size  $n_\ell \times n_\ell$  for  $\ell = 1, \dots, r$ . For  $\ell_1, \ell_2 \in \{1, \dots, r\}$ , define matrices  $\bar{B}_{\ell_1, \ell_2} = (\bar{L}_{ij})_{i \in \mathcal{N}_{\ell_1}, j \in \mathcal{N}_{\ell_2}} \in \mathbb{R}^{n_{\ell_1} \times n_{\ell_2}}$ , and  $B_{\ell_1, \ell_2}^* = (L_{ij}^*)_{i \in \mathcal{N}_{\ell_1}, j \in \mathcal{N}_{\ell_2}} \in \mathbb{R}^{n_{\ell_1} \times n_{\ell_2}}$ . Clearly,  $\bar{B}_{\ell, \ell} = \mathbf{E}_{n_\ell}$  for  $\ell \in \{1, \dots, r\}$ , and  $\bar{B}_{\ell_1, \ell_2} = \mathbf{0}_{n_{\ell_1} \times n_{\ell_2}}$  if  $\ell_1 \neq \ell_2$ , where  $\mathbf{E}_n \in \mathbb{R}^{n \times n}$  is a matrix of ones. For all

$1 \leq \ell_1, \ell_2 \leq r$ , define  $R_{\ell_1, \ell_2} := \|\bar{B}_{\ell_1, \ell_2} - B_{\ell_1, \ell_2}^*\|_F$ . Given a random graph corresponding to  $(n, \alpha)$ , for each algorithm ADMIPC, M-IALM( $1/\sqrt{n}$ ), and RPCAB, we compute five different statistics. The first three statistics are the maximum, minimum and average of  $\left\{\frac{R_{\ell, \ell}}{n_\ell}\right\}_{\ell \in \{1, \dots, r\}}$ , and are denoted by  $s_{\max}$ ,  $s_{\min}$ , and  $s_{\text{av}}$ , respectively. The fourth statistic is  $s_{\text{off}} := \frac{\sqrt{\sum_{(\ell_1, \ell_2): \ell_1 < \ell_2} R_{\ell_1, \ell_2}^2}}{\sqrt{\sum_{(\ell_1, \ell_2): \ell_1 < \ell_2} n_{\ell_1} n_{\ell_2}}}$ . These first four statistics show how close  $\bar{L}^*$  to the true clustering encoded by the BDO matrix  $\bar{L}$ . The fifth statistic  $s_f$  is about the fraction of clusters recovered correctly. For  $\ell \in \{1, \dots, r\}$ , define

$$(3.21) \quad E_\ell := \frac{R_{\ell, \ell}}{n_\ell}, \quad E_\ell^c := \frac{\sqrt{\sum_{t: t \neq \ell} R_{\ell, t}^2}}{\sqrt{\sum_{t: t \neq \ell} n_\ell n_t}}.$$

We call cluster  $\ell$  “recovered” if  $E_\ell < \tau_1$  and  $E_\ell^c < \tau_2$ . We set  $\tau_1 = 0.4$  and  $\tau_2 = 0.1$  for all three algorithms. Let  $\bar{r}$  denote the number of recovered clusters, i.e.,  $\bar{r} := |\{\ell : E_\ell < \tau_1, E_\ell^c < \tau_2\}|$ . The fifth statistic reported is  $s_f := \frac{\bar{r}}{r}$ . Note that all five statistics take values in  $[0, 1]$  interval.

Given  $(n, \alpha)$  and  $p_0 \in \{1, 0.9, 0.8\}$ , the underlying clustering of each 10 random graphs are estimated using ADMIPC, M-IALM, and RPCAB. Table 2, Table 3, and Table 4 report the averages of 5 statistics:  $s_{\max}$ ,  $s_{\min}$ ,  $s_{\text{av}}$ ,  $s_{\text{off}}$ ,  $s_f$  over the 10 instances for  $p_0 = 1$ ,  $p_0 = 0.9$ , and  $p_0 = 0.8$ , respectively. Numerical results show that increasing  $n$ , and/or decreasing  $\alpha$  adversely affect the performances of all three methods. However, the negative impact is more serious for M-IALM and RPCAB. Indeed, the results corresponding to  $s_f$  statistic show that while ADMIPC can detect more than %90 of clusters all the time,  $s_f$  values for M-IALM, and RPCAB decrease significantly (there are many instances for which  $s_f$  is 0) when  $n$  and  $\alpha$  change as discussed above. By investigating the results carefully, we see that usually the large values of  $E_\ell$  cause the failure of ADMIPC and M-IALM. But for RPCAB, the failure is mainly due to  $E_\ell^c > \tau_2$ . Although  $E_\ell^c$  are not reported, one can drive this result by comparing  $s_{\text{off}}$  values corresponding to different scenarios. Let  $\rho_0 = 1/\sqrt{n}$ . Indeed, all most all the time when  $\alpha < 0.9$  and/or  $n \geq 300$ , low-rank component of M-IALM( $\rho_0$ ) solution,  $L_{\rho_0}^*$ , violates  $\text{Tr}(L_{\rho_0}^*) = n$  condition, which causes  $E_\ell \approx 1$ , i.e.,  $(L_{\rho_0}^*)_{ij} \approx 0$  for all  $i, j \in \mathcal{N}_\ell$ , for all  $\ell$  such that  $n_\ell$  is small; hence, M-IALM( $\rho_0$ ) cannot detect small size clusters. To overcome this issue, Chen et al. [8] proposed bisection on  $\rho$ . Although this approach reduces the error  $E_\ell$  significantly, it causes some entries  $(L_\rho^*)_{ij} \approx 1$  such that  $i \in \mathcal{N}_{\ell_1}$ ,  $j \in \mathcal{N}_{\ell_2}$  and  $\ell_1 \neq \ell_2$ ; hence, merging two different clusters. Intuitively, the reason why our model

in (2.4) works better is that it considers both type of errors at the same time in a more tighter formulation than RPCA in (2.3).

Next, we compared cpu times required for ADMIPC and RPCAB to terminate on randomly generated networks as in Section 3.1 with  $n \in \{500, 1000\}$ ,  $\alpha = 0.95$  and  $p_0 = 0.9$ . For each  $n$ , we generated 5 instances; Table 5 displays the averages of **cpu**, **svd** and **iter<sub>B</sub>** statistics over 5 instances, where **cpu**, **svd** and **iter<sub>B</sub>** denote runtime (in seconds), total number of SVD computations, and the number of M-IALM calls within RPCAB, respectively.

### 3.3.2 ADMIPC vs Modularity Maximization

In this section, we compare ADMIPC with Louvain method [4]. Let  $\tau_d = 0.05$  and  $L^*$  be the optimal low-rank component computed by ADMIPC. If  $|L_{ii}^* - 1| > \tau_d$  for some  $i = 1, \dots, n$ , we declare failure; otherwise,  $T^* \in \mathbb{S}_n$  is constructed as follows:  $T_{ij}^* = 1$  if  $L_{ij}^* \geq \bar{\tau}$ , and  $T_{ij}^* = 0$  otherwise, where  $\bar{\tau} = 0.55$ . Based on  $T^*$ , we put nodes  $i$  and  $j$  in the same cluster if  $T_{ij}^* = 1$ . We compare the clusterings generated by ADMIPC, and Louvain method with the ground truth using three different measures of similarity: Jaccard index, normalized mutual information (NMI<sub>SG</sub>) using Strehl and Ghosh normalization [20], and portion of exactly recovered clusters (PERC). All three measures take values in  $[0, 1]$  interval, and values close to 1 correspond to desirable clusterings. Let  $\mathcal{G} = (\mathcal{N}, \mathcal{E})$  denote the network,  $\mathcal{C} = \{\mathcal{N}_i\}_{i=1}^r$ , which is a partition of  $\mathcal{N}$ , represent the ground truth, and  $\mathcal{C}' = \{\mathcal{N}'_j\}_{j=1}^{r'}$  represent the clustering computed by an algorithm.

1. **Jaccard index:** Let  $a$  be the number of node pairs that belong to the same clusters in both  $\mathcal{C}$  and  $\mathcal{C}'$ ,  $b$  be the number of pairs that are in the same cluster in  $\mathcal{C}$  but in different clusters in  $\mathcal{C}'$ , and  $c$  be the number of pairs that are in the same cluster in  $\mathcal{C}'$  but in different clusters in  $\mathcal{C}$ . The Jaccard’s index is defined as  $\frac{a}{a+b+c}$ . It has many applications in geology and ecology [23]; but it is a sensitive measure [17]. The Jaccard’s index is in  $[0, 1]$  interval. It is 1 when  $\mathcal{C}$  and  $\mathcal{C}'$  are exactly the same, and equal to 0 when there is no common pair classified in the same cluster in both  $\mathcal{C}$  and  $\mathcal{C}'$ .
2. **Normalized Mutual Information (NMI):** This measure is based on information theory, and it quantifies the reduction in our uncertainty about one cluster if we know the other one [22]. Define  $n_i := |\mathcal{N}_i|$  for  $i = 1, \dots, r$ ,  $n'_j := |\mathcal{N}'_j|$  for  $j = 1, \dots, r'$ , and  $m_{ij} = |\mathcal{N}_i \cap \mathcal{N}'_j|$  for all  $i, j$ . Then the mutual information  $\mathcal{I}(\mathcal{C}, \mathcal{C}') := \sum_{i=1}^r \sum_{j=1}^{r'} \frac{m_{ij}}{n} \log_2 \left( \frac{m_{ij}/n}{n_i n'_j / n^2} \right)$ . By normalizing



the mutual information, we force it to be between fixed ranges as well as improving its sensitivity [22, 24]. Let  $\mathcal{H}(\mathcal{C}) := -\sum_{i=1}^r \frac{n_i}{n} \log_2 \left( \frac{n_i}{n} \right)$  represent the entropy associated with clustering  $\mathcal{C}$  and  $\mathcal{H}(\mathcal{C}') := -\sum_{j=1}^{r'} \frac{n'_j}{n} \log_2 \left( \frac{n'_j}{n} \right)$  represent the entropy associated with clustering  $\mathcal{C}'$ . There are different ways of normalizing but we use the method introduced by Strehl and Ghosh [20]. In this method,  $\text{NMI}_{\text{SG}} = \frac{\mathcal{I}(\mathcal{C}, \mathcal{C}')}{\sqrt{\mathcal{H}(\mathcal{C})\mathcal{H}(\mathcal{C}')}} is between 0 and 1.$

3. **Portion of Exactly Recovered Clusters (PERC):** This measure is a secondary measure and we introduced it to make the comparison in case of tightness in the other measures. Let  $\bar{r}$  represent the total number of clusters that are both in  $\mathcal{C}$  and  $\mathcal{C}'$ , i.e. the number of clusters identified by the algorithm correctly. Then PERC is equal to  $\frac{\bar{r}}{r}$ .

For each  $n \in \{100, 200, 300, 400, 500\}$  random networks are generated as described in Section 3.1 for  $\alpha \in \{0.5, 0.6, \dots, 1\}$ . For each  $(n, \alpha)$  setting, we generated 20 random graphs, and compute two clusterings using ADMIPC and Louvain method. When  $p_0 < 1$ , if an edge is not observable, then we set the corresponding entry to 0 in the incidence matrix  $D$  for Louvain method. For a fixed  $(n, \alpha)$  setting and  $p_0 \in \{1, 0.9, 0.8\}$ , each of the three measures are evaluated on the 20 clusterings generated by ADMIPC on 20 random instances. The mean of these values are reported in Table 6, Table 7, and Table 8 for  $p_0 = 1$ ,  $p_0 = 0.9$ , and  $p_0 = 0.8$ , respectively. Note that the output of Louvain method depends on the initial ordering of the nodes. Hence, for the same graph, this method can generate different clusterings for different orderings. Therefore, for each 20 random graphs corresponding to fixed  $(n, \alpha)$ , we run Louvain method for 200 different ordering of nodes (generated randomly such that each ordering is equally likely). Numerical results show that our method outperforms Louvain method almost every time for all the three measures. It is important to note that increasing  $n$  adversely affects the performance of both methods. Moreover, for Louvain method, the clustering quality decreases significantly as  $\alpha$  decreases, i.e., the variation among the cluster sizes increases, while it is not the case for our method, and the clustering quality is not impacted significantly. Analyzing the results we find that for a fixed  $n$ , Louvain method tends to merge small clusters when  $\alpha$  is small. On the other hand, ADMIPC does not show any trend for the first two measures for changing  $\alpha$ , and almost every time correctly identifies even small size clusters. When  $\alpha$  is fixed and  $n$  increases, the clustering performance of Louvain method decreases again, which agrees with

the discussion on resolution limit. As  $n$  increases, the number of small size clusters increases in the ground truth as well, and more clusters are merged together by Louvain method, while the number of isolated nodes which originally belong to different clusters increases for ADMIPC. In summary, there are two key points which suggest that ADMIPC is more reliable than Louvain method. First, ADMIPC works well even for small values of  $\alpha$ . Second, by increasing  $n$ , the performance of both algorithms decrease: Louvain method tends to merge smaller clusters, while our algorithm generates some isolated nodes. But as discussed in [17], generating some isolated nodes is less severe than merging some clusters, which makes ADMIPC more reliable.

#### 4 Conclusion and Future Work

We proposed a convex optimization model for detecting nonoverlapping clusters in a partially observed undirected networks; and developed an ADMM algorithm to solve it. Since our formulation is tighter than the robust PCA formulation proposed in [8], we were able to find the true clustering even when the robust PCA formulation failed in our numerical tests. Moreover, our method is not sensitive to moderate changes in variance among cluster sizes, and on the randomly generated networks outperformed Louvain method, which maximizes the modularity and suffers from resolution limit. Extending our formulation to cluster *overlapping* communities in *weighted* networks is a potentially important future research direction. Due to limited space and time, we could not include computational results on real datasets; but they will soon be made available online at <http://www2.ie.psu.edu/aybat/codes.html>.

## References

- [1] N. S. AYBAT AND G. IYENGAR, *An alternating direction method with increasing penalty for stable principal component pursuit*. submitted to Computational Optimization and Applications, arXiv:1309.6553v2, 2013.
- [2] N. S. AYBAT, S. MA, AND D. GOLDFARB, *Efficient algorithms for robust and stable principal component pursuit problems*, Computational Optimization and Applications, (2014), pp. 1–29.
- [3] D. P. BERTSEKAS, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, 1982.
- [4] V.D. BLONDEL, J.L. GUILLAUME, R. LAMBIOTTE, AND E.L.J.S. MECH, *Fast unfolding of communities in large networks*, J. Stat. Mech, (2008), p. P10008.
- [5] S. BOYD, N. PARIKH, E. CHU, B. PELEATO, AND J. ECKSTEIN, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Found. Trends Mach. Learn., 3 (2011), pp. 1–122.
- [6] U. BRANDES, D. DELLING, M. GAERTLER, R. GORKE, M. HOEFER, Z. NIKOLOSKI, AND D. WAGNER, *On modularity clustering*, IEEE Transactions on Knowledge and Data Engineering, 20 (2008), pp. 172–188.
- [7] E. J. CANDÈS, X. LI, Y. MA, AND WRIGHT J., *Robust principle component analysis?*, Journal of ACM, 58 (2011), pp. 1–37.
- [8] Y. CHEN, A. JALALI, S. SANGHAVI, AND H. XU, *Clustering partially observed graphs via convex optimization*, Journal of Machine Learning Research, 15 (2014), pp. 2213–2238.
- [9] S. FORTUNATO, *Community detection in graphs*, Physics Reports, 486 (2010), pp. 75–174.
- [10] S. FORTUNATO AND M. BARTHÉLEMY, *Resolution limit in community detection*, Proceedings of the National Academy of Sciences, 104 (2007), pp. 36–41.
- [11] B. H. GOOD, Y. DE MONTJOYE, AND A. CLAUSET, *Performance of modularity maximization in practical contexts*, Phys. Rev. E, 81 (2010), p. 046106.
- [12] B. HE AND H. YANG, *Some convergence properties of a method of multipliers for linearly constrained monotone variational inequalities*, Operations Research Letters, 23 (1998), pp. 151–161.
- [13] B.S. HE, H. YANG, AND S.L. WANG, *Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities*, Journal of Optimization Theory and Applications, 106 (2000), pp. 337–356.
- [14] B. S. HE, L. Z. LIAO, D. R. HAN, AND H. YANG, *A new inexact alternating directions method for monotone variational inequalities*, Mathematical Programming, Series A, 92 (2002), pp. 103–118.
- [15] A. LANCICHINETTI AND S. FORTUNATO, *Limits of modularity maximization in community detection*, Phys. Rev. E, 84 (2011), p. 066122.
- [16] Z. LIN, M. CHEN, AND Y. MA, *The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices*. arXiv:1009.5055v3, 2013.
- [17] G. W. MILLIGAN AND D. A. SCHILLING, *Asymptotic and finite sample characteristics of four external criterion measures*, Multivariate Behavioral Research, 20 (1985), pp. 97–109.
- [18] M. E. J. NEWMAN, *Detecting community structure in networks*, EPJB, 38 (2004), pp. 321–330.
- [19] M. E. J. NEWMAN AND M. GIRVAN, *Finding and evaluating community structure in networks*, Phys. Rev. E, 69 (2004), p. 026113.
- [20] A. STREHL AND J. GHOSH, *Cluster ensembles - a knowledge reuse framework for combining multiple partitions*, J. Mach. Learn. Res., 3 (2003), pp. 583–617.
- [21] M. TAO AND X. YUAN, *Recovering low-rank and sparse components of matrices from incomplete and noisy observations*, SIAM Journal on Optimization, 21 (2011), pp. 57–81.
- [22] N. X. VINH, J. EPPS, AND J. BAILEY, *Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance*, J. Mach. Learn. Res., 11 (2010), pp. 2837–2854.
- [23] S. WAGNER AND D. WAGNER, *Comparing clusterings - an overview*, Tech. Report 2006-04, Universität Karlsruhe (TH), 2007.
- [24] J. WU, H. XIONG, AND J. CHEN, *Adapting the right measures for k-means clustering*, in Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, ACM, 2009, pp. 877–886.

$n$	$\alpha$	ADMIPC					M-IALM( $1/\sqrt{n}$ )					RPCAB				
		$s_{\max}$	$s_{\min}$	$s_{\text{av}}$	$s_{\text{off}}$	$s_f$	$s_{\max}$	$s_{\min}$	$s_{\text{av}}$	$s_{\text{off}}$	$s_f$	$s_{\max}$	$s_{\min}$	$s_{\text{av}}$	$s_{\text{off}}$	$s_f$
100	1	0.00	0.00	0.00	0.00	<b>1.00</b>	0.01	0.00	0.00	0.00	<b>1.00</b>	0.01	0.00	0.00	0.00	<b>1.00</b>
	0.9	0.00	0.00	0.00	0.00	<b>1.00</b>	0.02	0.00	0.00	0.00	<b>1.00</b>	0.02	0.00	0.00	0.00	<b>1.00</b>
	0.8	0.00	0.00	0.00	0.00	<b>1.00</b>	0.29	0.00	0.06	0.00	<b>0.90</b>	0.02	0.00	0.00	0.00	<b>1.00</b>
	0.7	0.02	0.00	0.00	0.00	<b>1.00</b>	1.00	0.00	0.25	0.00	<b>0.78</b>	0.04	0.02	0.02	0.02	<b>0.90</b>
	0.6	0.03	0.00	0.01	0.00	<b>1.00</b>	1.00	0.00	0.40	0.00	<b>0.60</b>	0.15	0.03	0.06	0.04	<b>0.80</b>
200	1	0.00	0.00	0.00	0.00	<b>1.00</b>	0.10	0.00	0.04	0.00	<b>1.00</b>	0.02	0.00	0.01	0.00	<b>1.00</b>
	0.9	0.01	0.00	0.00	0.00	<b>1.00</b>	1.00	0.00	0.36	0.00	<b>0.61</b>	0.02	0.00	0.00	0.00	<b>1.00</b>
	0.8	0.10	0.00	0.01	0.00	<b>1.00</b>	1.00	0.00	0.48	0.00	<b>0.50</b>	0.29	0.04	0.20	0.20	<b>0.00</b>
	0.7	0.28	0.00	0.05	0.00	<b>0.99</b>	1.00	0.00	0.58	0.00	<b>0.40</b>	0.35	0.02	0.18	0.20	<b>0.00</b>
	0.6	0.30	0.00	0.04	0.00	<b>0.98</b>	1.00	0.00	0.61	0.00	<b>0.40</b>	0.44	0.00	0.17	0.20	<b>0.00</b>
300	1	0.01	0.00	0.00	0.00	<b>1.00</b>	0.80	0.38	0.57	0.00	<b>0.06</b>	0.00	0.00	0.00	0.00	<b>1.00</b>
	0.9	0.10	0.00	0.01	0.00	<b>1.00</b>	1.00	0.00	0.57	0.00	<b>0.41</b>	0.28	0.08	0.20	0.20	<b>0.00</b>
	0.8	0.43	0.00	0.06	0.00	<b>0.96</b>	1.00	0.00	0.62	0.00	<b>0.38</b>	0.41	0.01	0.19	0.20	<b>0.00</b>
	0.7	0.37	0.00	0.05	0.00	<b>0.98</b>	1.00	0.00	0.67	0.00	<b>0.33</b>	0.55	0.00	0.17	0.20	<b>0.00</b>
	0.6	0.36	0.00	0.04	0.00	<b>0.97</b>	1.00	0.00	0.64	0.00	<b>0.36</b>	0.44	0.01	0.18	0.20	<b>0.00</b>
400	1	0.05	0.00	0.00	0.00	<b>1.00</b>	1.00	0.97	1.00	0.00	<b>0.00</b>	0.02	0.00	0.00	0.00	<b>1.00</b>
	0.9	0.22	0.00	0.02	0.00	<b>1.00</b>	1.00	0.00	0.65	0.00	<b>0.34</b>	0.31	0.01	0.19	0.20	<b>0.00</b>
	0.8	0.47	0.00	0.07	0.00	<b>0.95</b>	1.00	0.00	0.70	0.00	<b>0.30</b>	0.56	0.00	0.19	0.20	<b>0.00</b>
	0.7	0.49	0.00	0.08	0.00	<b>0.94</b>	1.00	0.00	0.69	0.00	<b>0.31</b>	0.56	0.00	0.22	0.20	<b>0.00</b>
	0.6	0.35	0.00	0.06	0.00	<b>0.98</b>	1.00	0.00	0.65	0.00	<b>0.33</b>	0.55	0.02	0.21	0.20	<b>0.00</b>
500	1	0.12	0.00	0.01	0.00	<b>1.00</b>	1.00	1.00	1.00	0.00	<b>0.00</b>	0.07	0.00	0.00	0.00	<b>1.00</b>
	0.9	0.31	0.00	0.04	0.00	<b>0.99</b>	1.00	0.00	0.70	0.00	<b>0.31</b>	0.36	0.01	0.19	0.20	<b>0.00</b>
	0.8	0.53	0.00	0.08	0.00	<b>0.90</b>	1.00	0.00	0.72	0.00	<b>0.29</b>	0.57	0.00	0.20	0.20	<b>0.00</b>
	0.7	0.47	0.00	0.08	0.00	<b>0.94</b>	1.00	0.00	0.65	0.00	<b>0.35</b>	0.54	0.01	0.22	0.20	<b>0.00</b>
	0.6	0.43	0.00	0.07	0.00	<b>0.94</b>	1.00	0.00	0.59	0.00	<b>0.42</b>	0.53	0.01	0.22	0.20	<b>0.00</b>

Table 2: The mean values for 5 statistics for  $p_0 = 1$

$n$	$\alpha$	ADMIPC					M-IALM( $1/\sqrt{n}$ )					RPCAB				
		$s_{\max}$	$s_{\min}$	$s_{\text{av}}$	$s_{\text{off}}$	$s_f$	$s_{\max}$	$s_{\min}$	$s_{\text{av}}$	$s_{\text{off}}$	$s_f$	$s_{\max}$	$s_{\min}$	$s_{\text{av}}$	$s_{\text{off}}$	$s_f$
100	1	0.00	0.00	0.00	0.00	<b>1.00</b>	0.04	0.00	0.01	0.00	<b>1.00</b>	0.04	0.00	0.01	0.00	<b>1.00</b>
	0.9	0.00	0.00	0.00	0.00	<b>1.00</b>	0.11	0.00	0.03	0.00	<b>1.00</b>	0.03	0.00	0.01	0.00	<b>1.00</b>
	0.8	0.00	0.00	0.00	0.00	<b>1.00</b>	0.72	0.00	0.16	0.00	<b>0.82</b>	0.00	0.00	0.00	0.00	<b>1.00</b>
	0.7	0.03	0.00	0.01	0.00	<b>1.00</b>	1.00	0.00	0.34	0.00	<b>0.64</b>	0.07	0.00	0.02	0.00	<b>1.00</b>
	0.6	0.08	0.00	0.02	0.00	<b>1.00</b>	1.00	0.00	0.40	0.00	<b>0.60</b>	0.21	0.05	0.11	0.10	<b>0.50</b>
200	1	0.00	0.00	0.00	0.00	<b>1.00</b>	0.39	0.05	0.18	0.00	<b>0.94</b>	0.00	0.00	0.00	0.00	<b>1.00</b>
	0.9	0.01	0.00	0.00	0.00	<b>1.00</b>	1.00	0.00	0.48	0.00	<b>0.51</b>	0.02	0.00	0.00	0.00	<b>1.00</b>
	0.8	0.18	0.00	0.03	0.00	<b>0.99</b>	1.00	0.00	0.53	0.00	<b>0.45</b>	0.32	0.12	0.21	0.19	<b>0.00</b>
	0.7	0.33	0.00	0.06	0.00	<b>0.98</b>	1.00	0.00	0.61	0.00	<b>0.40</b>	0.39	0.06	0.21	0.19	<b>0.00</b>
	0.6	0.42	0.00	0.06	0.01	<b>0.93</b>	1.00	0.00	0.66	0.00	<b>0.34</b>	0.65	0.02	0.25	0.19	<b>0.00</b>
300	1	0.03	0.00	0.00	0.00	<b>1.00</b>	1.00	0.75	0.93	0.00	<b>0.00</b>	0.02	0.00	0.00	0.00	<b>1.00</b>
	0.9	0.19	0.00	0.03	0.00	<b>1.00</b>	1.00	0.00	0.65	0.00	<b>0.33</b>	0.32	0.12	0.21	0.19	<b>0.00</b>
	0.8	0.40	0.00	0.08	0.00	<b>0.97</b>	1.00	0.00	0.67	0.00	<b>0.33</b>	0.47	0.02	0.22	0.19	<b>0.00</b>
	0.7	0.44	0.00	0.07	0.01	<b>0.95</b>	1.00	0.00	0.69	0.00	<b>0.31</b>	0.62	0.00	0.22	0.19	<b>0.00</b>
	0.6	0.32	0.00	0.05	0.00	<b>0.99</b>	1.00	0.00	0.64	0.00	<b>0.36</b>	0.55	0.00	0.22	0.19	<b>0.00</b>
400	1	0.10	0.00	0.01	0.00	<b>1.00</b>	1.00	1.00	1.00	0.00	<b>0.00</b>	0.06	0.00	0.00	0.00	<b>1.00</b>
	0.9	0.33	0.00	0.06	0.00	<b>0.98</b>	1.00	0.00	0.71	0.00	<b>0.30</b>	0.39	0.10	0.23	0.19	<b>0.00</b>
	0.8	0.49	0.00	0.10	0.01	<b>0.94</b>	1.00	0.00	0.71	0.00	<b>0.30</b>	0.49	0.01	0.20	0.20	<b>0.00</b>
	0.7	0.56	0.00	0.11	0.00	<b>0.90</b>	1.00	0.00	0.69	0.00	<b>0.31</b>	0.60	0.00	0.23	0.19	<b>0.00</b>
	0.6	0.45	0.00	0.09	0.00	<b>0.95</b>	1.00	0.00	0.67	0.00	<b>0.33</b>	0.55	0.00	0.23	0.19	<b>0.00</b>
500	1	0.19	0.00	0.02	0.00	<b>1.00</b>	1.00	1.00	1.00	0.00	<b>0.00</b>	0.08	0.00	0.01	0.00	<b>1.00</b>
	0.9	0.39	0.00	0.05	0.00	<b>0.98</b>	1.00	0.00	0.74	0.00	<b>0.26</b>	0.48	0.08	0.24	0.19	<b>0.00</b>
	0.8	0.51	0.00	0.10	0.00	<b>0.92</b>	1.00	0.00	0.74	0.00	<b>0.25</b>	0.70	0.00	0.24	0.19	<b>0.00</b>
	0.7	0.54	0.00	0.10	0.00	<b>0.91</b>	1.00	0.00	0.69	0.00	<b>0.31</b>	0.62	0.01	0.25	0.19	<b>0.00</b>
	0.6	0.54	0.00	0.09	0.00	<b>0.93</b>	1.00	0.00	0.65	0.00	<b>0.33</b>	0.63	0.00	0.25	0.19	<b>0.00</b>

Table 3: The mean values for 5 statistics for  $p_0 = 0.9$

$n$	$\alpha$	ADMIPC					M-IALM( $1/\sqrt{n}$ )					RPCAB				
		$s_{\max}$	$s_{\min}$	$s_{\text{av}}$	$s_{\text{off}}$	$s_f$	$s_{\max}$	$s_{\min}$	$s_{\text{av}}$	$s_{\text{off}}$	$s_f$	$s_{\max}$	$s_{\min}$	$s_{\text{av}}$	$s_{\text{off}}$	$s_f$
100	1	0.00	0.00	0.00	0.00	<b>1.00</b>	0.10	0.00	0.04	0.00	<b>1.00</b>	0.01	0.00	0.00	0.00	<b>1.00</b>
	0.9	0.00	0.00	0.00	0.00	<b>1.00</b>	0.29	0.00	0.08	0.00	<b>0.96</b>	0.00	0.00	0.00	0.00	<b>1.00</b>
	0.8	0.02	0.00	0.00	0.00	<b>1.00</b>	1.00	0.00	0.28	0.00	<b>0.72</b>	0.03	0.00	0.01	0.00	<b>1.00</b>
	0.7	0.02	0.00	0.00	0.00	<b>1.00</b>	1.00	0.00	0.40	0.00	<b>0.60</b>	0.09	0.03	0.05	0.04	<b>0.80</b>
	0.6	0.15	0.00	0.04	0.00	<b>1.00</b>	1.00	0.00	0.43	0.00	<b>0.60</b>	0.29	0.11	0.18	0.15	<b>0.20</b>
200	1	0.01	0.00	0.00	0.00	<b>1.00</b>	0.88	0.35	0.64	0.00	<b>0.07</b>	0.00	0.00	0.00	0.00	<b>1.00</b>
	0.9	0.11	0.00	0.02	0.00	<b>1.00</b>	1.00	0.00	0.58	0.00	<b>0.42</b>	0.16	0.03	0.06	0.03	<b>0.80</b>
	0.8	0.30	0.00	0.06	0.00	<b>0.99</b>	1.00	0.00	0.61	0.00	<b>0.40</b>	0.37	0.14	0.23	0.18	<b>0.00</b>
	0.7	0.37	0.00	0.09	0.01	<b>0.96</b>	1.00	0.00	0.63	0.00	<b>0.37</b>	0.46	0.09	0.25	0.18	<b>0.00</b>
	0.6	0.50	0.00	0.09	0.01	<b>0.92</b>	1.00	0.00	0.70	0.00	<b>0.30</b>	0.61	0.02	0.24	0.18	<b>0.00</b>
300	1	0.10	0.00	0.02	0.00	<b>1.00</b>	1.00	0.99	1.00	0.00	<b>0.00</b>	0.05	0.00	0.01	0.00	<b>1.00</b>
	0.9	0.26	0.00	0.06	0.00	<b>1.00</b>	1.00	0.00	0.72	0.00	<b>0.29</b>	0.34	0.14	0.22	0.17	<b>0.00</b>
	0.8	0.62	0.00	0.15	0.01	<b>0.85</b>	1.00	0.00	0.70	0.00	<b>0.30</b>	0.64	0.12	0.28	0.18	<b>0.00</b>
	0.7	0.57	0.00	0.10	0.01	<b>0.89</b>	1.00	0.00	0.73	0.00	<b>0.27</b>	0.61	0.00	0.24	0.19	<b>0.00</b>
	0.6	0.42	0.00	0.08	0.01	<b>0.94</b>	1.00	0.00	0.65	0.00	<b>0.36</b>	0.65	0.00	0.26	0.17	<b>0.00</b>
400	1	0.17	0.00	0.04	0.00	<b>1.00</b>	1.00	1.00	1.00	0.00	<b>0.00</b>	0.14	0.03	0.06	0.03	<b>0.80</b>
	0.9	0.36	0.00	0.06	0.00	<b>0.99</b>	1.00	0.00	0.77	0.00	<b>0.22</b>	0.45	0.14	0.25	0.17	<b>0.00</b>
	0.8	0.63	0.00	0.14	0.01	<b>0.87</b>	1.00	0.00	0.74	0.00	<b>0.26</b>	0.67	0.01	0.26	0.20	<b>0.00</b>
	0.7	0.58	0.00	0.11	0.01	<b>0.90</b>	1.00	0.00	0.70	0.00	<b>0.30</b>	0.69	0.00	0.27	0.18	<b>0.00</b>
	0.6	0.57	0.00	0.12	0.01	<b>0.89</b>	1.00	0.00	0.67	0.00	<b>0.33</b>	0.71	0.06	0.29	0.18	<b>0.00</b>
500	1	0.28	0.00	0.04	0.00	<b>1.00</b>	1.00	1.00	1.00	0.00	<b>0.00</b>	0.29	0.14	0.22	0.16	<b>0.00</b>
	0.9	0.43	0.00	0.07	0.00	<b>0.98</b>	1.00	0.00	0.79	0.00	<b>0.20</b>	0.57	0.13	0.27	0.18	<b>0.00</b>
	0.8	0.66	0.00	0.13	0.01	<b>0.91</b>	1.00	0.00	0.76	0.00	<b>0.25</b>	0.66	0.00	0.22	0.21	<b>0.00</b>
	0.7	0.54	0.00	0.11	0.00	<b>0.92</b>	1.00	0.00	0.69	0.00	<b>0.31</b>	0.63	0.02	0.25	0.18	<b>0.00</b>
	0.6	0.53	0.00	0.11	0.00	<b>0.91</b>	1.00	0.00	0.67	0.00	<b>0.33</b>	0.64	0.06	0.28	0.17	<b>0.00</b>

Table 4: The mean values for 5 statistics for  $p_0 = 0.8$

$n$	instance #	ADMIPC							RPCAB							
		$s_{\max}$	$s_{\min}$	$s_{av}$	$s_{\text{off}}$	$s_f$	cpu	svd	$s_{\max}$	$s_{\min}$	$s_{av}$	$s_{\text{off}}$	$s_f$	cpu	svd	iter <sub>B</sub>
500	1	0.21	0.00	0.04	0.00	1.00	<b>3.8</b>	<b>30</b>	0.30	0.15	0.22	0.18	0.00	<b>13.6</b>	<b>61</b>	3
	2	0.17	0.00	0.03	0.00	1.00	<b>3.8</b>	<b>31</b>	0.27	0.12	0.21	0.18	0.00	<b>13.3</b>	<b>58</b>	3
	3	0.16	0.00	0.02	0.00	1.00	<b>3.9</b>	<b>33</b>	0.27	0.10	0.20	0.18	0.00	<b>13.4</b>	<b>58</b>	3
	4	0.20	0.00	0.02	0.00	1.00	<b>3.4</b>	<b>29</b>	0.31	0.12	0.21	0.18	0.00	<b>13.3</b>	<b>60</b>	3
	5	0.19	0.00	0.03	0.00	1.00	<b>3.5</b>	<b>31</b>	0.33	0.14	0.22	0.18	0.00	<b>13.8</b>	<b>60</b>	3
1000	1	0.57	0.00	0.05	0.01	0.98	<b>26.7</b>	<b>40</b>	0.55	0.00	0.27	0.22	0.00	<b>159.3</b>	<b>92</b>	4
	2	0.37	0.00	0.03	0.00	1.00	<b>25.7</b>	<b>37</b>	0.50	0.00	0.26	0.22	0.00	<b>163.2</b>	<b>92</b>	4
	3	0.31	0.00	0.04	0.00	1.00	<b>25.9</b>	<b>39</b>	0.59	0.00	0.27	0.22	0.00	<b>162.2</b>	<b>92</b>	4
	4	0.34	0.00	0.04	0.01	1.00	<b>27.2</b>	<b>41</b>	0.45	0.00	0.26	0.22	0.00	<b>161.5</b>	<b>90</b>	4
	5	0.52	0.00	0.04	0.00	0.98	<b>26.0</b>	<b>38</b>	0.58	0.00	0.26	0.22	0.00	<b>162.9</b>	<b>93</b>	4

Table 5: The cpu times, total svd numbers, and 5 statistics for  $p_0 = 0.9$  and  $\alpha = 0.95$

		Louvain					ADMIPC				
		$\frac{n}{\alpha}$	100	200	300	400	500	100	200	300	400
Jaccard Index	1	99.9	98.9	95.5	90.0	84.1	100	100	100	100	99.9
	0.9	99.8	93.9	78.2	67.4	58.8	100	100	100	99.9	99.8
	0.8	99.5	83.9	73.4	67.9	66.4	100	99.9	99.9	99.9	99.9
	0.7	98.2	82.0	78.5	76.7	76.8	100	99.9	99.9	99.9	99.9
	0.6	94.3	86.1	86.1	86.9	86.8	99.9	99.9	99.9	99.9	99.9
	0.5	94.3	88.7	89.9	88.9	87.9	100	99.9	99.9	99.9	94.9
NMI <sub>SG</sub>	1	100	99.8	99.3	98.6	97.9	100	100	100	100	99.9
	0.9	99.9	98.6	94.5	91.2	88.1	100	100	100	99.9	99.8
	0.8	99.8	93.7	88.9	86.5	85.5	100	99.9	99.9	99.7	99.7
	0.7	99.0	89.4	87.2	86.2	86.2	100	99.8	99.7	99.7	99.7
	0.6	95.6	88.0	87.9	88.0	87.8	99.9	99.8	99.8	99.8	99.8
	0.5	93.1	87.2	88.0	86.8	86.2	100	99.8	99.8	99.9	99.9
PERC	1	99.9	98.8	95.0	89.1	82.6	100	100	100	100	99.8
	0.9	99.7	90.1	51.8	32.1	18.1	100	100	100	98.5	94.8
	0.8	99.2	49.7	20.3	9.95	6.30	100	99.5	98.6	86.0	88.4
	0.7	92.9	27.4	9.50	8.37	8.38	100	96.5	88.6	86.4	85.0
	0.6	65.3	15.0	10.5	10.7	10.3	99	94	90.8	91.6	91.1
	0.5	42.5	15.3	13.1	12.3	13.4	100	93.7	93.3	93.3	86.1

Table 6: The mean values for 3 measures in % when  $p_0 = 1$

		Louvain					ADMIPC				
		$\frac{n}{\alpha}$	100	200	300	400	500	100	200	300	400
Jaccard Index	1	99.9	99.1	96.5	92.8	85.0	100	100	100	100	99.9
	0.9	99.9	95.0	78.4	67.4	59.7	100	100	99.9	99.9	99.7
	0.8	99.7	85.0	73.6	68.7	67.5	100	99.9	99.9	99.9	99.9
	0.7	98.4	83.4	78.1	77.8	77.3	99.9	99.9	99.9	99.9	99.9
	0.6	93.6	85.9	86.8	87.0	86.9	99.9	99.9	99.9	99.9	99.9
	0.5	94.6	90.2	90.6	88.7	88.1	100	99.9	99.9	99.9	99.9
NMI <sub>SG</sub>	1	99.9	99.8	99.5	99.0	98.1	100	100	100	100	99.9
	0.9	99.9	98.8	94.4	91.1	88.1	100	100	99.9	99.9	99.6
	0.8	99.9	93.9	88.6	86.5	85.8	100	99.9	99.8	99.6	99.6
	0.7	99.1	89.9	87.0	86.4	86.3	99.9	99.7	99.6	99.7	99.7
	0.6	95.2	87.7	88.1	87.9	87.9	99.9	99.6	99.7	99.8	99.8
	0.5	93.3	88.2	88.5	86.6	86.3	100	99.7	99.8	99.8	99.9
PERC	1	99.9	99.0	96.2	92.2	83.3	100	100	100	100	99.4
	0.9	99.9	91.1	51.8	30.9	17.2	100	100	99.3	97.5	90.2
	0.8	99.5	50.1	17.1	10.3	5.73	100	98	93.3	81.8	84.2
	0.7	93.3	26.0	7.97	7.40	7.51	99.0	93.0	87.0	84.1	82.9
	0.6	63.5	13.0	9.83	9.31	10.0	99	90.5	87.5	91.2	87.6
	0.5	43.2	16.6	13.8	11.5	13.1	100	91.2	90.5	92.7	90

Table 7: The mean values for 3 measures in % when  $p_0 = 0.9$

		Louvain					ADMIPC				
		$\frac{n}{\alpha}$	100	200	300	400	500	100	200	300	400
Jaccard Index	1	99.9	99.4	97.4	93.7	86.5	100	100	100	100	99.8
	0.9	99.9	95.9	78.9	66.4	60.2	100	100	99.9	99.7	99.4
	0.8	99.8	85.1	73.20	67.5	67.4	100	99.8	99.8	99.8	99.8
	0.7	97.9	82.3	79.3	75.7	75.4	99.9	99.8	99.9	99.9	99.9
	0.6	93.4	87.1	87.5	87.6	87.1	99.9	99.9	99.9	99.9	99.9
	0.5	94.3	88.5	89.2	88.8	88.5	99.9	99.9	99.9	99.9	99.9
NMI <sub>SG</sub>	1	100	99.9	99.6	99.1	98.3	100	100	100	100	99.9
	0.9	99.9	99.0	94.6	90.7	87.9	100	100	99.9	99.7	99.2
	0.8	99.9	93.7	88.4	86.1	85.3	100	99.8	99.6	99.4	99.4
	0.7	98.8	89.3	87.1	85.7	85.6	99.9	99.7	99.5	99.5	99.6
	0.6	95.1	88.3	88.2	88.1	87.8	99.8	99.6	99.5	99.7	99.7
	0.5	93.1	87.2	87.5	86.5	86.3	99.8	99.7	99.8	99.7	99.8
PERC	1	99.9	99.3	97.2	93.2	85.1	100	100	100	100	98.8
	0.9	99.9	91.8	52.1	29.9	14.42	100	100	99	92.7	80.6
	0.8	99.8	45.4	16.9	8.88	5.26	100	96.5	89.3	78.1	78.6
	0.7	91.4	24.6	8.17	6.61	6.73	99	93	84	77.6	81.1
	0.6	64.8	14.7	9.61	8.58	10.08	98	89	80.4	86.2	84.6
	0.5	42.7	14.3	12.0	10.7	12.1	97	89.3	88.8	87.7	90

Table 8: The mean values for 3 measures in % when  $p_0 = 0.8$